GPU-accelerated implicit discontinuous Galerkin approximation of hypersonic flows

Sebastien Terrana*, Dominique Hoskin†, Jan Eichstädt‡, Ngoc-Cuong Nguyen§, and Jaime Peraire¶
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139

High-order discontinuous Galerkin (DG) methods have emerged as an attractive approach for large eddy simulation of turbulent flows owing to their high accuracy and implicit dissipation properties. However, the application of DG methods for hypersonic flows is still challenging due to the high-computational cost and the lack of robust shock capturing algorithms. In this paper, we address the efficiency and robustness of Discontinuous Galerkin methods. To that end we develop a high-order implicit discontinuous Galerkin method for the numerical simulation of hypersonic flows on graphics processors (GPUs). The main ingredients in our approach include: (*i*) implicit high-order DG approximation on unstructured/adapted meshes, (*ii*) shock capturing for hypersonic flows, (*iii*) iterative solution methods with CUDA/MPI implementation on GPU clusters, and (*iv*) effective matrix-free preconditioner with reduced basis approximation of the Jacobian matrix. Numerical results on several test cases are presented to validate our method.

I. Introduction

Many critical decisions in the design of hypersonic vehicles require the ability to predict accurately surface skin friction and aerodynamic heating which, in turn, depend on complex physical processes such as shock wave/boundary layer interaction, boundary layer separation, and laminar-to-turbulent transition. A deep understanding of the hypersonic boundary layers, including boundary layer transition, is crucial to the design of thermal protection systems and flight control for hypersonic vehicles. Gaining this understanding is often hampered by the difficulty to obtain high quality data to validate the theory and modeling efforts.

The use of second-order finite volume schemes is almost ubiquitous in computational fluid dynamics for hypersonic flows. These schemes are well-suited for the solution the RANS equations. However, it has been recently argued [1] that high-order numerical schemes are better suited for large eddy simulation of transitional and turbulent flows. Indeed, the prediction of transitional and turbulent flows relies on accurate wave propagation for which high-order accuracy is known to be key. High-order finite difference, finite volume and discontinuous Galerkin methods have been developed for hypersonic problems. Among them, discontinuous Galerkin methods offer several advantages. First, DG methods are based on a strong mathematical foundation that can be exploited for error estimation and mesh adaptation. Also, they provide local conservation, a stable discretization of the convective operator, and are well-suited for turbulence simulations due to the ab initio separation of scales in the variational formulation. Most importantly, DG methods allow for high-order discretizations on complex geometries and unstructured meshes; which is a critical feature to simulate transitional and turbulent flows over the complex three-dimensional geometries commonly encountered in industrial applications. Successful numerical predictions, e.g. [2, 3], further encourage the use of DG methods for LES.

Over the past years, we have developed a class of hybridized DG (HDG) methods [4–12] for aerodynamic applications, which possess important advantages over other DG methods [13–15] in terms of both convergence rate and computational complexity. The solver described in [3] is implicit in time and solves the nonlinear system at each timestep by using a Newton-–Krylov—Schwarz solver with block ILU factorization [16]. While this solver has been demonstrated to scale up 100,000 cores, it requires the formation and memory storage of both the Jacobian matrix and the block ILU factorization. As such, it does not scale well on modern GPU clusters. As such, it does not scale well on modern GPU clusters. In order to be able to scale up the code and solve larger problems, we have developed and implemented a

*Postdoctoral Associate, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139. Email: terrana@mit.edu.

†PhD Student, Department of Aeronautics and Astronautics, MIT

‡PhD Student, Department of Aeronautics, Imperial College of Science Technology and Medicine, London SW7 2AZ, UK

§Principal Research Scientist, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139. Email: cuongng@mit.edu. Senior AIAA member.

¶Professor, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 02139. Email: peraire@mit.edu. AIAA Fellow.

matrix-free implicit DG solver on GPUs using CUDA and MPI. To solve the large linear systems of equations generated at each timestep, the DG solver is equipped with a matrix-free preconditioner using the reduced basis method and low rank updates to the approximate Jacobian [17].

It turns out that most existing shock capturing approaches developed for DG methods are unsuitable for unsteady hypersonic flows . Existing approaches based on adding dissipation in the presence of under-resolved features are unable to distinguish between under-resolution due to shocks and under resolution due to the presence of turbulent structures. We have developed a novel shock capturing method for LES [18, 19]. The proposed method relies on physical viscosities to resolve and smooth sharp under-resolved flow features that may otherwise lead to numerical instability, such as shock waves and sharp thermal and shear layers. To that end, we devise various sensors to detect when and where the shear viscosity, bulk viscosity and thermal conductivity of the fluid do not suffice to stabilize the numerical solution. In such cases, the fluid viscosities are selectively increased to ensure the cell Péclet number is of order one so that these flow features can be well represented with the available grid resolution. Numerical results show the shock capturing method performs robustly from transonic to hypersonic regimes, provides sharp shock profiles, and has no detrimental effect on the resolved turbulent structures. These three features are critical to enable robust and accurate LES of shock flows.

In this paper, we present an implicit high-order discontinuous Galerkin (DG) method for the numerical simulation of hypersonic transitional flows at high Reynolds numbers. The main ingredients in our approach include: (*i*) implicit high-order DG approximation on unstructured/adapted meshes, (*ii*) shock capturing for hypersonic flows, (*iii*) iterative solution methods with CUDA/MPI implementation on GPU clusters, and (*iv*) effective matrix-free preconditioner with reduced basis approximation of the Jacobian matrix. We present numerical results to demonstrate and validate our method for several hypersonic flows.

II. Numerical Methodology

A. Governing equations

Let $t_f > 0$ be a final time and let $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$ be an open, connected and bounded physical domain with Lipschitz boundary $\partial\Omega$. The unsteady compressible Navier-Stokes equations in conservation form are given by

$$\boldsymbol{q} - \nabla \boldsymbol{u} = 0\,, \qquad \text{in } \Omega \times [0, t_f)\,, \tag{1a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{u}) + \nabla \cdot \boldsymbol{G}(\boldsymbol{u}, \boldsymbol{q}) = 0\,, \qquad \text{in } \Omega \times [0, t_f)\,, \tag{1b}$$

$$\boldsymbol{B}(\boldsymbol{u}, \boldsymbol{q}) = 0\,, \qquad \text{on } \partial\Omega \times [0, t_f)\,, \tag{1c}$$

$$\boldsymbol{u} - \boldsymbol{u}_0 = 0\,, \qquad \text{on } \Omega \times \{0\}\,. \tag{1d}$$

were $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$ is the computational domain with bound boundary $\partial\Omega$. Here, $\boldsymbol{u} = (\rho, \rho v_j, \rho E)$, $j = 1, ..., d$ is the m-dimensional ($m = d + 2$) vector of conserved quantities, $\boldsymbol{u}_0$ is an initial state, $\boldsymbol{B}(\boldsymbol{u}, \boldsymbol{q})$ is a boundary operator, and $\boldsymbol{F}(\boldsymbol{u})$ and $\boldsymbol{G}(\boldsymbol{u}, \boldsymbol{q})$ are the inviscid and viscous fluxes of dimensions $m \times d$,

$$\boldsymbol{F}(\boldsymbol{u}) = \begin{pmatrix} \rho v_j \\ \rho v_i v_j + \delta_{ij} p \\ v_j(\rho E + p) \end{pmatrix}, \qquad \boldsymbol{G}(\boldsymbol{u}, \boldsymbol{q}) = -\begin{pmatrix} 0 \\ \tau_{ij} \\ v_i \tau_{ij} - f_j \end{pmatrix}, \qquad i, j = 1, \ldots, d\,, \tag{2}$$

where p denotes the thermodynamic pressure, τ_{ij} the viscous stress tensor, f_j the heat flux, and δ_{ij} is the Kronecker delta. For a calorically perfect gas in thermodynamic equilibrium, $p = (\gamma - 1)\left(\rho E - \rho \|\boldsymbol{v}\|^2/2\right)$, where $\gamma = c_p/c_v > 1$ is the specific heat ratio and in particular $\gamma \approx 1.4$ for air. Here c_p and c_v denote the specific heats at constant pressure and volume, respectively. For a Newtonian fluid with the Fourier's law of heat conduction, the viscous stress tensor and heat flux are given by

$$\tau_{ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right) + \beta\,\frac{\partial v_k}{\partial x_k}\delta_{ij}, \qquad\qquad f_j = -\,\kappa\,\frac{\partial T}{\partial x_j}, \tag{3}$$

where T denotes temperature, μ the dynamic (shear) viscosity, β the bulk viscosity, $\kappa = c_p\,\mu/Pr$ the thermal conductivity, and Pr the Prandtl number. In particular, $Pr \approx 0.71$ for air, and additionally $\beta = 0$ under the Stokes' hypothesis.

B. Shock capturing

To deal with shocks, we extend the physics-based artificial viscosity approach introduced by Fernandez et al. [18]. This approach relies on shock, thermal gradient and shear sensors. The *shock sensor* is constructed such that

$$s_\beta(\mathbf{x}) = s_\theta \cdot s_\omega, \qquad s_\theta = -\frac{h_\beta}{k}\frac{\nabla\cdot\mathbf{u}}{a^*}, \qquad s_\omega = \frac{(\nabla\cdot\mathbf{u})^2}{(\nabla\cdot\mathbf{u})^2 + |\nabla\times\mathbf{u}|^2 + \varepsilon}, \tag{4}$$

where ε is a constant of the order of the machine precision squared, k is the polynomial degree and a^* is the critical speed of speed of sound. The element size is taken along the direction of the density gradient

$$h_\beta(\mathbf{x}) = h_{\text{ref}}\frac{|\nabla\rho|}{\sqrt{\nabla\rho^T\cdot\mathbf{M}_h^{-1}\nabla\rho + \varepsilon}} \tag{5}$$

where $\mathbf{M}_h$ is the metric tensor of the mesh, and h_{ref} is the size of the reference element used in the construction of $\boldsymbol{M}_h$. The *shear sensor* is also designed to detect under-resolved features, namely velocity gradients, and is constructed from

$$s_\mu(\mathbf{x}) = \frac{h_{\text{ref}}}{k}\frac{\|\boldsymbol{\mathcal{L}}(\mathbf{u})\cdot\mathbf{x}_{\boldsymbol{\xi}}^T\|_2}{u_{\max}} \tag{6}$$

where $\|\cdot\|_2$ denotes the usual Euclidean norm, $u_{\max}$ is the maximum isentropic velocity (obtained by reversibly converting all the energy into kinetic energy)

$$u_{\max} = \sqrt{|\mathbf{u}|^2 + \frac{2}{\gamma-1}a^2}, \qquad \boldsymbol{\mathcal{L}}(\mathbf{u}) = \frac{\partial\mathbf{u}_i}{\partial\mathbf{x}_j}\left(1-\delta_{ij}\right), \tag{7}$$

with δ_{ij} representing Kronecker's delta.

We then add artificial bulk viscosity and artificial molecular viscosity (β^*, μ^*) to the physical values such that:

$$\beta = \beta_f + \beta^*, \qquad \mu = \mu_f + \mu^*,$$

where the amount of viscosities are determined to ensure a cell Péclet number of $O(1)$ as follows

$$\beta^*(\mathbf{x}) = \hat{s}_\beta\rho\frac{k_\beta h_\beta}{k}\sqrt{|\mathbf{u}|^2 + a^{*2}} \tag{8a}$$

$$\mu^*(\mathbf{x}) = \hat{s}_\mu\rho\frac{k_\mu h_\mu}{k}\sqrt{|\mathbf{u}|^2 + a^{*2}}. \tag{8b}$$

Here $k_{\mu,\kappa} = 1$, $k_\beta = 1.5$, and $\left(\hat{s}_\beta, \hat{s}_\mu\right)$ denote the smoothly bounded values of the sensors in equations (4) and (6). We also note that

$$h_\mu(\mathbf{x}) = h_{\text{ref}}\inf_{|\mathbf{a}|=1}\left\{\mathbf{a}^T\cdot\mathbf{M}_h\,\mathbf{a}\right\}. \tag{9a}$$

Finally, a smoothing operator is applied to (β^*, μ^*) to make them C^0 continuous. Since at most moderately high accuracy orders are used for the numerical examples in this paper, we employ an element-wise linear reconstruction procedure analogous to that introduced in [20] for the element size.

C. Discontinuous Galerkin method

The DG discretization of the unsteady compressible Navier-Stokes equations reads as follows: Find $\left(\boldsymbol{q}_h(t), \boldsymbol{u}_h(t)\right) \in \boldsymbol{Q}_h^k \times \boldsymbol{\mathcal{V}}_h^k$ such that

$$\left(\boldsymbol{q}_h, \boldsymbol{r}\right)_{\mathcal{T}_h} + \left(\boldsymbol{u}_h, \nabla\cdot\boldsymbol{r}\right)_{\mathcal{T}_h} - \left\langle\widehat{\boldsymbol{u}}_h, \boldsymbol{r}\cdot\boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h} = 0\,, \tag{10a}$$

$$\left(\frac{\partial\,\boldsymbol{u}_h}{\partial t}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left(\boldsymbol{F}(\boldsymbol{u}_h) + \boldsymbol{G}(\boldsymbol{u}_h, \boldsymbol{q}_h), \nabla\boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle\widehat{\boldsymbol{f}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h) + \widehat{\boldsymbol{g}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h, \boldsymbol{q}_h), \boldsymbol{w}\right\rangle_{\partial\mathcal{T}_h} = 0\,, \tag{10b}$$

for all $(\boldsymbol{r}, \boldsymbol{w}, \boldsymbol{\mu}) \in \boldsymbol{Q}_h^k \times \boldsymbol{\mathcal{V}}_h^k \times \boldsymbol{\mathcal{M}}_h^k$ and all $t \in [0, t_f)$, as well as

$$\left(\boldsymbol{u}_h|_{t=0} - \boldsymbol{u}_0, \boldsymbol{w}\right)_{\mathcal{T}_h} = 0\,, \tag{10c}$$

for all $\boldsymbol{w} \in \boldsymbol{\mathcal{V}}_h^k$. The finite element spaces and inner products above are described in [3]. Here $\widehat{\boldsymbol{f}}_h$ and $\widehat{\boldsymbol{g}}_h$ are the inviscid and viscous numerical fluxes which are defined on the interior faces as

$$\widehat{\boldsymbol{f}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h) = \boldsymbol{F}(\widehat{\boldsymbol{u}}_h) \cdot \boldsymbol{n} + \boldsymbol{\sigma}(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h; \boldsymbol{n}) \cdot (\boldsymbol{u}_h - \widehat{\boldsymbol{u}}_h)\,, \tag{11a}$$

$$\widehat{\boldsymbol{g}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h, \boldsymbol{q}_h) = \boldsymbol{G}(\widehat{\boldsymbol{u}}_h, \boldsymbol{q}_h) \cdot \boldsymbol{n}\,, \tag{11b}$$

and $\boldsymbol{n}$ is the unit normal vector pointing outwards from the elements. Note that this form of the numerical flux does not involve an explicit Riemann solver. Instead, it is the so-called stabilization matrix $\boldsymbol{\sigma}(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h; \boldsymbol{n}) \in \mathbb{R}^{m \times m}$ that implicitly defines the Riemann solver in DG methods. In this paper, we set $\boldsymbol{\sigma} = \lambda_{max}(\widehat{\boldsymbol{u}}_h)\, \boldsymbol{I}_m$, where λ_{max} denotes the maximum-magnitude eigenvalue of $\boldsymbol{A}_n = [\partial \boldsymbol{F} / \partial \boldsymbol{u}] \cdot \boldsymbol{n}$ and $\boldsymbol{I}_m$ is the $m \times m$ identity matrix, and which leads to a Lax-Friedrichs-type Riemann solver. Additional details on this stabilization matrix and the resulting Riemann solver are presented in [3, 10].

It remains to define the numerical trace. Different choices of the numerical trace yield different DG methods. For the local DG (LDG) method [21] the numerical trace is defined as $\widehat{\boldsymbol{u}}_h = \frac{1}{2}(\boldsymbol{u}_h^+ + \boldsymbol{u}_h^-) + (\boldsymbol{u}_h^+ \boldsymbol{\beta} \cdot \boldsymbol{n}^+ + \boldsymbol{u}_h^- \boldsymbol{\beta} \cdot \boldsymbol{n}^-)$ on the interior faces, where $\boldsymbol{\beta}$ is a vector-valued function. Note that $\boldsymbol{u}_h^+ = \boldsymbol{u}_h|_{F \in K^+}$ and $\boldsymbol{u}_h^- = \boldsymbol{u}_h|_{F \in K^-}$ denote the restriction of the numerical solution $\boldsymbol{u}_h$ on interior face F shared by elements K^+ and K^-. On the boundary faces, the definition of the numerical trace depends on the boundary conditions. The hybridized DG (HDG) method [3, 10] does not define the numerical trace $\widehat{\boldsymbol{u}}_h$ terms of the numerical solution. In the HDG method, the numerical trace becomes an additional variable to be solved by introducing another equation that imposes the continuity of the numerical flux $\widehat{\boldsymbol{f}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h) + \widehat{\boldsymbol{g}}_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{u}_h, \boldsymbol{q}_h)$ and enforces boundary conditions. In this paper, the LDG method is used to solve the compressible Navier-Stokes equations since it is suited to a matrix-free iterative solver discussed in the next section.

Finally, the semi-discrete system (10) is further discretized in time using high-order, L-stable diagonally implicit Runge-Kutta (DIRK) schemes [22]. The use of high-order, L-stable methods for the temporal discretization is important for accuracy and robustness when dealing with turbulent shock flows. Also, the use of implicit time integration schemes allows to examine the impact of the shock capturing method on the conditioning of the spatial discretization (10) through the ease of solving the nonlinear system of equations arising from the time discretization.

D. Solution method

The Newton's method is used to solve the nonlinear system of equations resulting from the temporal discretization of the system (10). In order to reduce the number of Newton iterations, we compute the initial guess $\boldsymbol{u}_h^{n,0}$ at n^{th} timestep by solving the following least-squares problem [3]:

$$\boldsymbol{u}_h^{n,0} := \sum_{j=1}^{J} \alpha_j\, \boldsymbol{u}_h^{n-j}$$

where $\boldsymbol{u}_h^l$ denotes the solution at the l^{th} timestep, and

$$(\alpha_1, \ldots, \alpha_J) = \arg \min_{(\beta_1, \ldots, \beta_J) \in \mathbb{R}^J} \left\| \boldsymbol{R}_{\text{NS}}\left(\sum_{j=1}^{J} \beta_j\, \boldsymbol{u}_h^{n-j} \right) \right\|. \tag{12}$$

This optimization problem is solved by using the Levenberg–Marquardt (LM) algorithm [23], where the gradient vectors $\partial \boldsymbol{R}_{\text{NS}} / \partial \beta_j$ are approximated by finite differences. This advanced initialization adds very little to the overall computational cost while reducing the number of Newton iterations compared to the standard initialization using just the previous solution.

In each Newton iteration, we use GMRES to solve the resulting linear system. In order to accelerate the convergence rate of GMRES we develop a matrix-free scalable preconditioner. We briefly describe one of our approaches to preconditioning that will scale linearly with the problem size. The key idea in devising a matrix-free preconditioner lies in the construction of an approximation to the Jacobian matrix $\boldsymbol{J}(\boldsymbol{u}_n)$ of dimension $N \times N$ through a suitable low-rank

approximation. Given the mass matrix $\boldsymbol{M}$ and a low-rank matrix $\boldsymbol{W}_n$ of dimension $N \times m$ with $m \ll N$ consisting of the previous solution vectors, our preconditioner has the following form:

$$\boldsymbol{P}_n = \boldsymbol{M} + \boldsymbol{V}_n \boldsymbol{D}_n^{-1} \boldsymbol{W}_n \,, \tag{13}$$

where $\boldsymbol{V}_n$ and $\boldsymbol{D}_n$ are chosen to satisfy the following approximation property:

$$\boldsymbol{P}_n \boldsymbol{W}_n = \boldsymbol{J}(\boldsymbol{u}_n) \boldsymbol{W}_n \,. \tag{14}$$

In order to satisfy this equation, we choose $\boldsymbol{D}_n = \boldsymbol{W}_n^T \boldsymbol{W}_n$ and $\boldsymbol{V}_n = \boldsymbol{J}(\boldsymbol{u}_n)\boldsymbol{W}_n - \boldsymbol{M}\boldsymbol{W}_n$. Using the Sherman–Morrison–Woodbury formula, we compute the inverse of the preconditioner $\boldsymbol{P}_n$ as:

$$\boldsymbol{P}_n^{-1} = \boldsymbol{M}^{-1} - \boldsymbol{M}^{-1} \boldsymbol{V}_n \left(\boldsymbol{D}_n + \boldsymbol{W}_n^T \boldsymbol{M}^{-1} \boldsymbol{V}_n \right)^{-1} \boldsymbol{W}_n^T \boldsymbol{M}^{-1} \,, \tag{15}$$

The low-rank preconditioner of the form (13) can be viewed as a generalization of the BFGS update [24–27] with a distinctive feature that our approach allows for arbitrary-rank approximation, whereas the BFGS update is only a rank-two approximation of the Jacobian matrix.

The computation of $\boldsymbol{J}(\boldsymbol{u}_n)\boldsymbol{W}_n$ can be expensive if we have to form the Jacobian matrix $\boldsymbol{J}(\boldsymbol{u}_n)$ and perform matrix-matrix multiplication. Instead, the product of the Jacobian matrix with any vector $\boldsymbol{y}$ can be approximately computed by the Taylor expansion as follows

$$\boldsymbol{J}(\boldsymbol{u}_n)\, \boldsymbol{y} \approx \frac{\boldsymbol{R}_{\mathrm{NS}}(\boldsymbol{u}_n + \epsilon \boldsymbol{y}) - \boldsymbol{R}_{\mathrm{NS}}(\boldsymbol{y})}{\epsilon} \,, \tag{16}$$

for small enough ϵ. Furthermore, since we only need to compute the first column of the matrix $\boldsymbol{J}(\boldsymbol{u}_n)\boldsymbol{W}_n$, it allows us to construct the preconditioner with only *one evaluation* of the residual vector. With this reduced preconditioning technique, the construction of the preconditioner adds very little cost to the preconditioned accelerated first-order methods described earlier. Therefore the computational complexity of our preconditioned GMRES is comparable to that without preconditioning.

E. CUDA/MPI implementation

The proposed discretization schemes and solution methods are implemented in the CUDA-accelerated multiphysics simulations (CAMPS) code, which is written using the C++ programming language with MPI-based parallelization and CUDA for GPUs. Because computing the residual vector is the most expensive operation in all simulations, it must be optimized. The residual vector is computed using Gauss quadrature which involves matrix-matrix multiplication of the form $\boldsymbol{R}_{\mathrm{NS}}(\boldsymbol{u}_n) = \boldsymbol{S} \times \boldsymbol{G}(\boldsymbol{u}_n)$, where $\boldsymbol{S}$ is a small matrix (related to the shape functions and their derivatives at the quadrature points on the master element) and $G(\boldsymbol{u}_n)$ is a very wide matrix. We optimize the residual calculation on GPUs by implementing a number of different algorithms to compute this matrix-matrix multiplication. In particular, tensor-product with sum-factorization is one of the implemented algorithms, while other algorithms involve the customized allocation of GPU's shared memory depending on the size of the matrix S. We use automatic tuning to pick the fastest algorithm depending on the problem dimension, the polynomial degree, the element type, and the particular GPUs. For parallel simulations, we divide the computational domain into subdomains and each GPU is responsible for computing the part of the residual vector on its own subdomain. MPI communication across neighboring subdomains is overlapped with the computation of the residual on the interior elements of each subdomain.

While the scaling of an application over large distributed memory systems is important, it does not assess how efficient the underlying hardware is actually utilized. In order to assess this additional performance aspect of our code, we present the achieved percentage of peak FLOP/s and memory bandwidth. We have chosen a Nvidia Titan V GPU, that is of the same generation as V100 GPU. The Titan V GPU has a theoretical peak performance of 7.45 TFLOP/s and a maximum bandwidth of 652GB/s. For the benchmarks on a multicore CPU system, we have chosen an Intel Xeon E5 2660 v3 CPU with 20 threads. This CPU has a theoretical peak performance of 528 GFLOP/s and a maximum bandwith of 68GB/s.

As a test case we have selected the canonical Taylor-Green Vortex problem. We execute this case with our flow solver for the polynomial degrees from $p = 1$ to $p = 5$ on hexahedral elements. The number of elements is changed between 27000 for $p = 1$ and 1000 for $p = 5$ to give the same number of degrees of freedom. The code is always executed to the same nondimensional simulation time. All cases are further executed with the same CFL number and the same tolerance threshold for the Newton and the GMRES(20) iterations. This results in a fair comparison for all polynomial degrees.

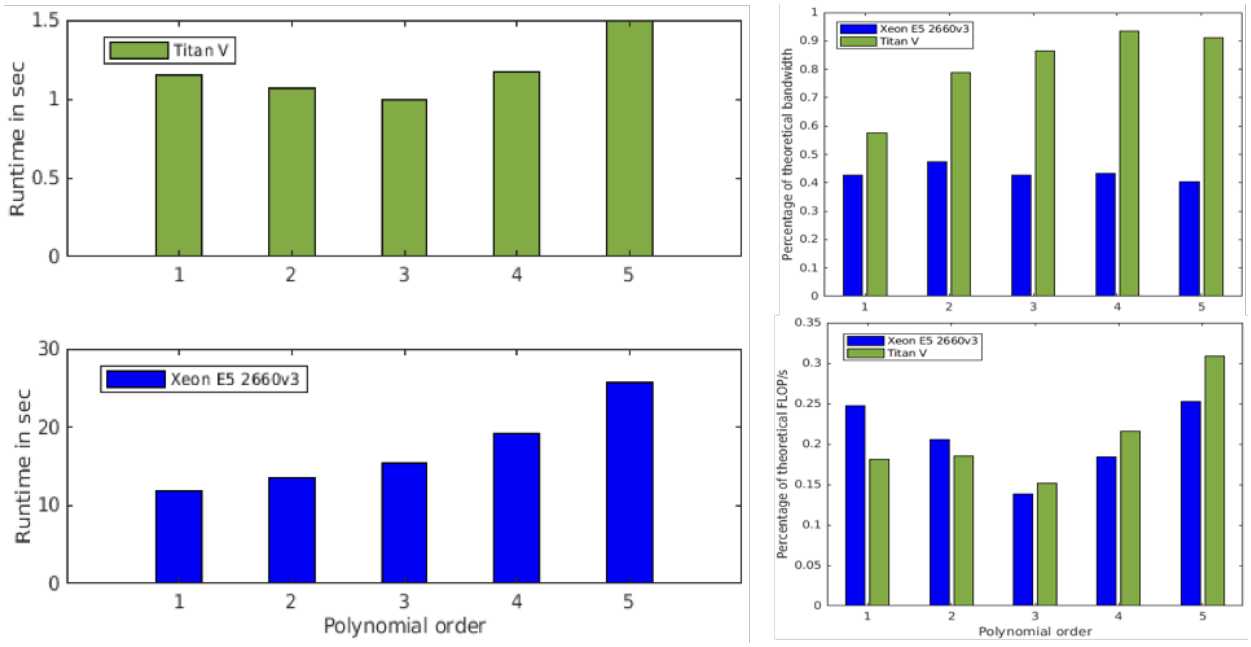

Fig. 1 ***Runtime comparison between CPU and GPU for different polynomial degrees.***

In Figure 1, we present scaled run times for both the employed CPU and GPU. The results show a very significant performance gain for the Titan V GPU, that varies between a factor of 16 for $p = 4$ and 12.5 for $p = 2$. The intrinsic characteristics of our algorithms result in increasing operation counts for higher polynomial orders, while on the other hand low polynomial orders are compromised by a less efficient memory access. Additionally, the run times are influenced by the complex interplay of element order with the number of iterations for the iterative solvers. All effects combined, the most time-efficient polynomial order on the GPU is $p = 3$, while for the CPU it is $p = 1$.

We also calculate the weighted averages of achieved FLOP/s and memory bandwidth of all kernels in the code that together comprise at least 80% of the run times. In all cases, by far the most important kernel with a share of 25% to 41% is the NVIDIA cuBLAS `dgemm` kernel, which is a highly optimized library on NVIDIA GPUs. The relative share of the `dgemm` kernel between polynomial orders varies and hence directly influence the overall numbers, as presented in Figure 1. For the Volta GPU our code is generally memory bandwidth limited, with a utilization rate of about 90%. For the lowest polynomial orders, the smaller array sizes lead to less ideal memory access patterns and hence lower bandwidth. For the CPU, the achieved bandwidth is around 45% of the theoretical maximum, indicating there is no clear performance bottleneck.

Figure 2 shows the weak and strong scaling of CAMPS for more than 1000 Nvidia Tesla V100 GPUs at the OLCF's Summit supercomputer through our recent Director's Discretionary Allocation. The results correspond to the three-dimensional simulation of the Edney Type IV shock/shock interaction problem. The numerical discretization is based on third-order DG and DIRK schemes. One MPI rank per GPU was used. The scaling tests were designed so that the work per GPU is the same as in the production runs for the project. The various runs differ on the number of elements across the spanwise extent of the computational domain. The two-dimensional mesh on the periodic plane and the time-step size are kept constant in all runs. For the strong scaling assessment, a fully resolved grid of approximately 3.2 million fourth-order elements was partitioned in approximately 30, 60, 120, 480, 960, and 1920 subdomains.

Excellent weak scaling is observed in Figure 2 up to 144 GPUs. In particular, performance degrades by about 5% when scaling from 36 to 144 GPUs. Furthermore, good weak scaling is observed up to 1152 GPUs as performance degrades by about 50% when increasing from 36 to 1152 GPUs. Similar observations can be made for the strong scaling test. Specifically, we obtain excellent strong scaling up to 240 GPUs and the scaling performance degrades as the number of GPUs increases. This is due to the increased number of iterations required to converge in the linear solution. This potential issue could be addressed by using one subdomain per computing node (OLCF's Summit has 6 V100 GPUs per node), as opposed to one subdomain per GPU. This would further extend the ideal scaling regime of the code by drastically increasing the number of elements per subdomain and taking advantage of the fast intra-node GPU-to-GPU communication.

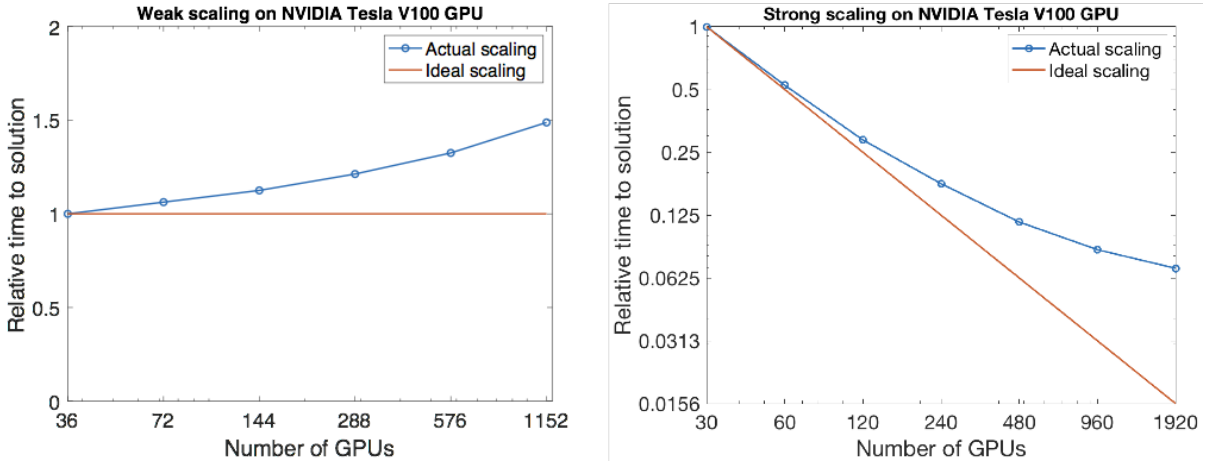

Fig. 2 ***Weak and strong scaling tests of our CAMPS code on NVIDIA Tesla V100 GPUs at the OLCF's Summit supercomputer.***

III. Results

A. Steady laminar hypersonic flows over a cylinder

Figure 3 shows a 2D steady simulation of a $M = 17.6$, $Re = 376,000$ flow around cylinder and demonstrates the ability of our shock capturing approach to deal with strong bow shocks. This problem was studied by Gnoffo and White [28] comparing the structured code LAURA and the unstructured code FUN3D. For this simulation, the cylinder is isothermal with wall temperature $T_{\text{wall}} = 500^{\text{o}}$ K and $p = 3$ polynomials on a grid of 100×250 quadrilateral elements have been used for the spatial approximation. Our simulation results agree well with those obtained using the LAURA code.

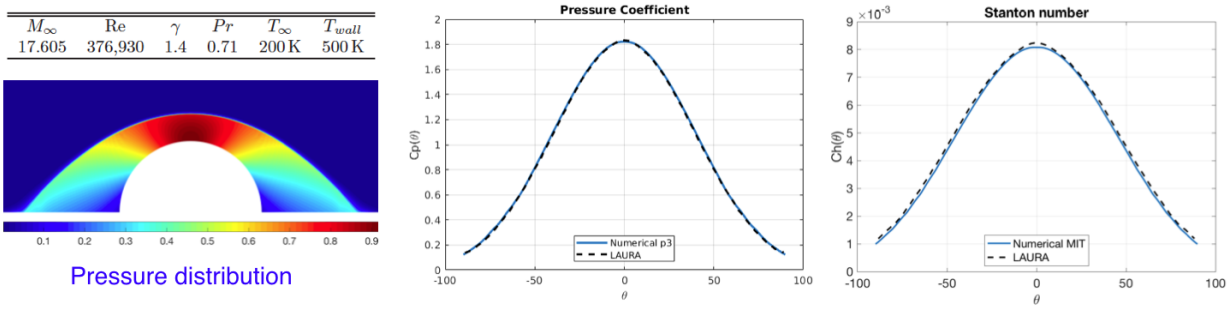

Fig. 3 2D Laminar hypersonic flow over a cylinder. The shock is captured within one element without oscillations using third order polynomials. The pressure field (top) and the pressure and heat transfer coefficients on the cylinder are shown and compared to the solution in [28].

B. Type IV Shock-shock interaction

Type IV Shock-shock interaction results in a very complex flow field with high pressure and heat flux peak in localized region. It occurs when the incident shock impinges on a bow shock and results in the formation of a supersonic impinging jet, a series of shock waves, expansion waves, and shear layers in a local area of interaction. The supersonic impinging jet, which is bounded by two shear layers separating the jet from the upper and lower subsonic regions, impinges on the body surface, and is terminated by a jet bow shock just ahead of the surface. This impinging jet bow shock wave creates a small stagnation region of high pressure and heating rates. Meanwhile, shear layers are formed to separate the supersonic jet from the lower and upper subsonic regions. Due to the complicated flow interactions, the flow is unsteady and periodic. The unsteadiness was confirmed by the experiment by Wieting and Holden [29].

In Figure 4, we show the computation of a Type IV shockwave interaction for free stream Mach number of 8.03

and $Re = 3.88 \times 10^5$, using $p = 2$ polynomials on a grid of 64, 300 quadrilateral elements. The left part of the figure shows the Mach number distribution over the computational domain. The middle portion of the figure consists of zooms of the Mach and temperature distributions in the interaction region, showing the formation of a supersonic jet that impinges the cylinder wall at a near right angle. Finally, the surface and heat transfer distributions on the cylinder surface compared to the experimental results reported in [29] are shown on the right. Note that, as a result of the shock interaction, the maximum surface heating rate is an order of magnitude higher than if the impinging shock wasn't there. This is a 2D simulation and it shows flow unsteadiness when sufficient resolution is used. The disagreement in the heating rate between the experimental data and the simulation in the region below the stagnation point is also found by other researchers. In order to assess the possibility of this disagreement being caused by high frequency 3D unsteady effects, we have performed a 3D simulation on a domain obtained by extruding the 2D domain and have found the time averaged results agree with those of the 2D simulation.

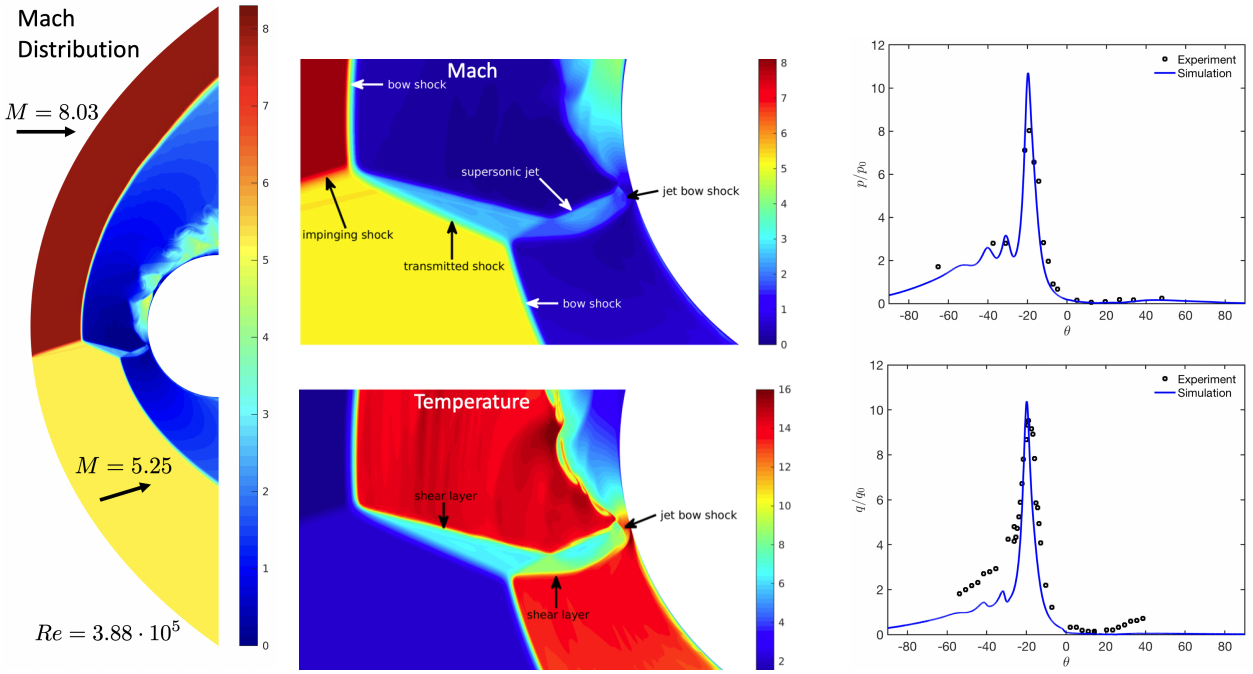

Fig. 4 Type IV shockwave interaction for an isothermal cylinder at $M_\infty = 08.03$ and a $Re = 3.88 \times 10^5$. The Mach number distribution is shown on the left. Details of the Mach and temperature distributions in the interaction region are shown in the middle and the surface pressure and heat transfer distributions are shown and compared to the experimental data by Wieting and Holden [29] on the right.

C. Steady laminar hypersonic flows over a compression ramp

The hypersonic flow over a compression ramp is a geometrically simple test case with experimental data for validation [30]. Flow conditions and geometry are shown in Figure 5. Figure 5 also shows the computation of a laminar hypersonic flow over a ramp. The displacement thickness of the boundary layer creates a weak oblique shock at the flat plate leading edge. A stronger oblique shock exists near the wedge corner. Additionally, the significant pressure rise behind the oblique shock at the wedge creates a strong adverse pressure gradient in the boundary layer. This adverse pressure gradient induces separation and creates a noticeable region of recirculating flow at the wedge corner. Finally, the laminar boundary layer reattachment region corresponds to a rise in local heating rates. The negative skin friction near the corner indicates the the flow separates. We observe that the computed skin friction and Stanton number coefficients agree well with the experimental data obtained by Holden [30]. We have computed this solution with quadratic and cubic approximations and the results show very little difference, thus indicating grid convergence.

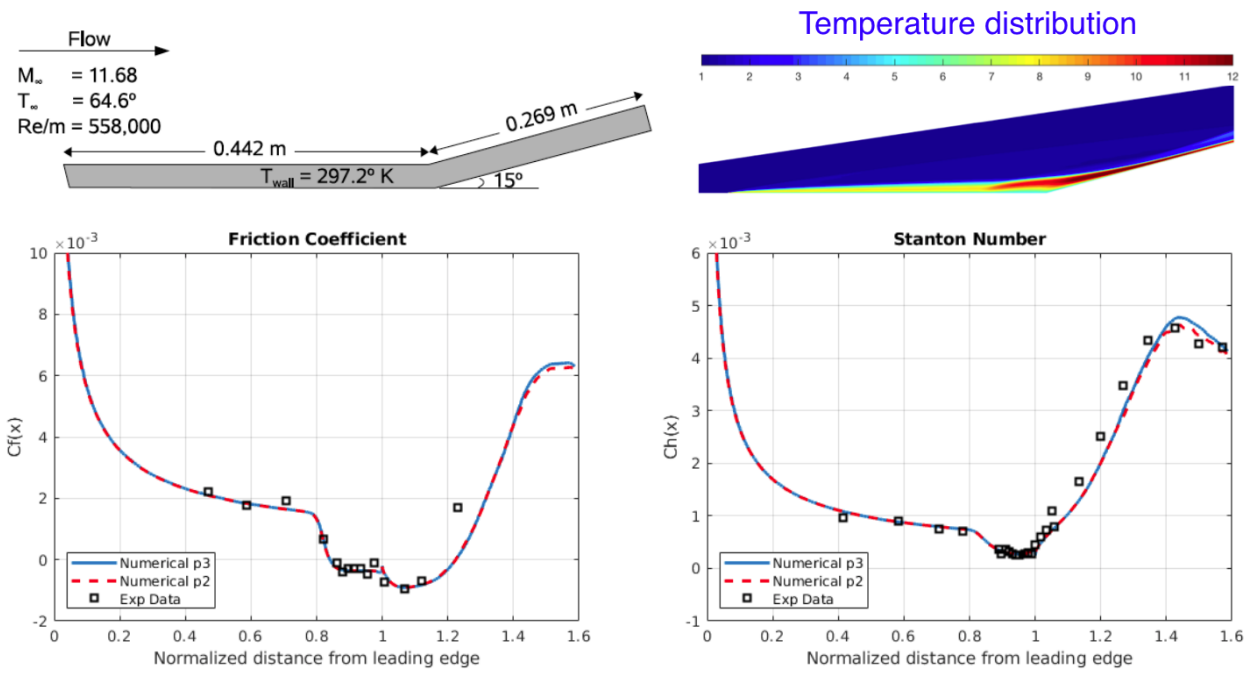

Fig. 5 Laminar $M = 11.68$ hypersonic flow over a ramp modelled as an isothermal wall. The figure shows the temperature field as well as the skin friction and Stanton number coefficients for quadratic and cubic approximations in comparison with the experimental data obtained by Holden [30].

D. Turbulent hypersonic flows over a double ramp

To illustrate the effects of flow transition on aerothermal heating, we compare a laminar flow with a transitional flow and observe the differences in heat transfer rates. The problem considered is a double-wedge at Mach 8.1 which has been experimentally studied in [31]. In the experiment, the boundary layer is observed to transition on the second ramp at about $x = 0.2$m measured from the leading edge. We compute a two solutions: the first solution is laminar, in the second solution, we simulate the effects of transition and turbulence by using a Spalart–Allmaras RANS model and prescribe transition at the transition location observed experimentally. Figure 6 shows a schematic of the problem together with a Schilieren image of the experiment, in the top row. The middle row shows the pressure and temperature distribution fields, which look visually similar for both simulations. For the transition computation, the flow transitions from laminar to turbulence over the separation bubble. The heat flux over the second ramp predicted by the transition computation is much higher than that predicted by the laminar computation. Compared to experimental data [31], the computation with prescribed transition shows good overall agreement.

E. Hypersonic flows over a 2D scramjet inlet

For scramjet vehicles, inaccurate prediction of drag and heat transfer coefficients may lead to mission failures. These coefficients increase rapidly and attain maximum values in laminar–to–turbulent transition regions. Therefore, understanding, predicting and controlling flow transition in hypersonic boundary layers plays a vital role in the design of scramjets. Transition mechanisms are influenced by several parameters related to flow conditions, free-stream disturbances, surface roughness, and geometries. For small disturbances, we usually observe natural transition that can conceptually be divided into three stages including receptivity, linear amplification of the dominant instability mode, and nonlinear breakdown to turbulence [33]. For large enough disturbances, nonlinear breakdown phenomena are immediately observed and the so-called bypass transition occurs rapidly. In addition to the second-mode and crossflow instabilities, another important instability in scramjet geometry is separation-induced transition, where a laminar boundary layer separates under the influence of adverse pressure gradients and transition develops within the separated shear layer. For multi–ramp inlets, separation-induced instability is likely to occur due to the formation of separation bubbles at the ramp corners.

We perform a 2D laminar computation of hypersonic flow over the SWL scramjet intake shown in Figure 7. This

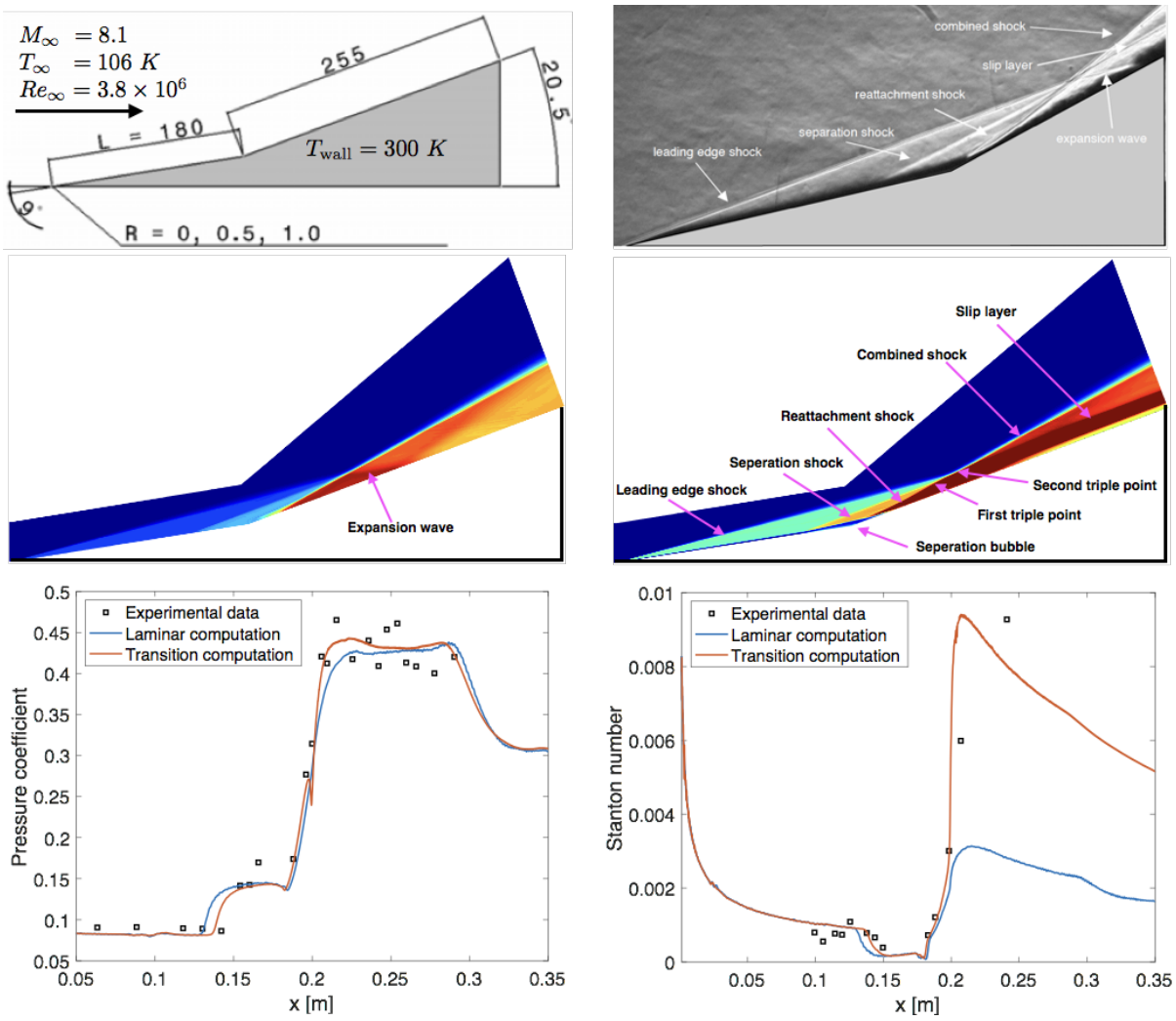

Fig. 6 ***Effects of transition on a double ramp flow: (top) geometry/flow conditions [31] and Schlieren image of the flow field [32], (middle) computed solution with prescribed transition, (bottom) comparison between computation with prescribed transition and laminar computation against experimental data [31].***

problem was studied by Frauholz et al. in [34]. We present in Figure 8 the computed results in comparison with the experimental data. We observe that the computed pressure coefficient is in reasonable agreement with the experimental data, while the computed Stanton number does not match the experimental one after the reattachment location of the first bubble on the second ramp. The mismatch between computation and experiment can be attributed to modeling error since our 2D laminar computation does not capture transition and turbulence phenomena, as well as geometry error since the SWL scramjet intake is a complex three-dimensional geometry with two end walls. We are currently performing 3D LES simulations to investigate transition and turbulence in the SWL scramjet intake.

IV. Conclusions

We have presented a matrix-free implicit discontinuous Galerkin solver and artificial-viscosity shock capturing for hypersonic flows on graphics processors. Our high-order DG solver was applied to a number of test cases. The computed results were compared to those simulation results from other codes as well as the experimental data. While good agreements were observed for two-dimensional flows, some discrepancy were noticeable for unsteady three-dimensional flows such as the Edney type IV interaction and the SWL scramjet inlet. We are currently investigating 3D large eddy simulations of these flows to assess and understand the physics behind hypersonic boundary layer transition.

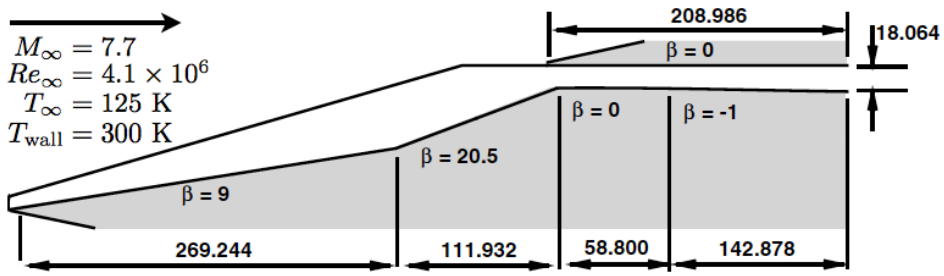

Fig. 7 ***Geometry in millimeter scale and flow conditions of the SWL intake [34].***

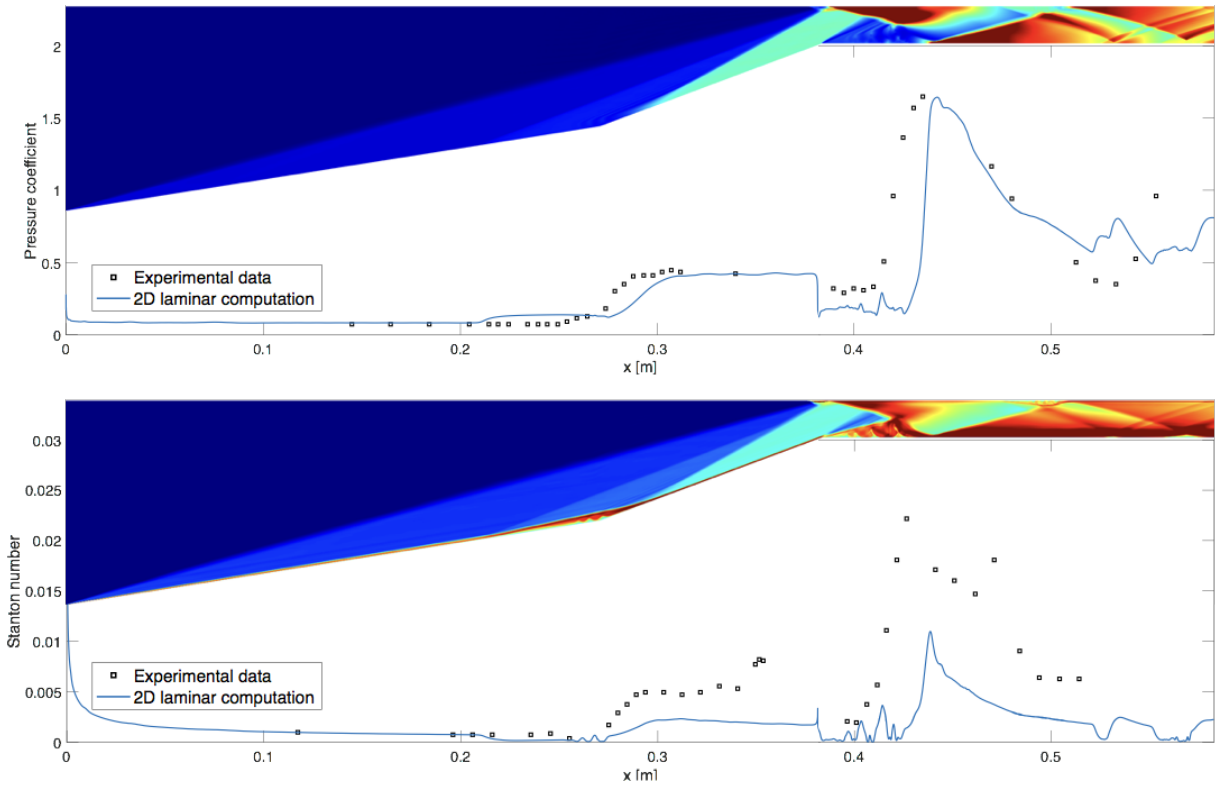

Fig. 8 ***2D laminar computation of hypersonic flow over the SWL intake [34]: Pressure coefficient distribution (top) and Stanton number distribution (bottom).***

Acknowledgments

The authors would like to thank Professor Wesley Harris from MIT for encouraging this work. They also gratefully acknowledge the AFOSR (under grant number FA9550-16-1-0214) for supporting this work. Jan Eichstädt acknowledges the support provided by a Fullbright Fellowship. Dominique Hoskin acknowledges the support of the Harris Provost Funds. Also, the authors thank the Barcelona Supercomputing Center and the Oak Ridge Leadwership Computing Facility for providing access to their GPU clusters.

References

[1] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D. L., Gropp, W., Lurie, E., and Mavriplis, D. J., "CFD vision 2030 study : A path to revolutionary computational aerosciences," Tech. Rep. NASA-CR-2014-218178, NASA Langley, 2014.

[2] Beck, A. D., Bolemann, T., Flad, D., Frank, H., Gassner, G. J., Hindenlang, F., and Munz, C.-D., "High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 76, No. 8, 2014, pp. 522–548. doi:10.1002/fld.3943, URL `http://doi.wiley.com/10.1002/fld.3943`.

[3] Fernandez, P., Nguyen, N. C., and Peraire, J., "The hybridized discontinuous Galerkin method for implicit large-eddy simulation of transitional turbulent flows," *Journal of Computational Physics*, Vol. 336, 2017, pp. 308–329. doi:10.1016/j.jcp.2017.02.015.

[4] Fernandez, P., Christophe, A., Terrana, S., Nguyen, N. C., and Peraire, J., "Hybridized discontinuous Galerkin methods for wave propagation," *Journal of Scientific Computing*, Vol. 77, No. 3, 2018, pp. 1566–1604. doi:10.1007/s10915-018-0811-x, URL http://link.springer.com/10.1007/s10915-018-0811-x.

[5] Moro, D., Nguyen, N. C., and Peraire, J., "Navier-Stokes Solution Using Hybridizable Discontinuous Galerkin methods," Tech. rep., Honolulu, Hawaii, jun 2011. doi:10.2514/6.2011-3407, URL http://arc.aiaa.org/doi/abs/10.2514/6.2011-3407.

[6] Nguyen, N. C., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 230, No. 4, 2011, pp. 1147–1170. doi:10.1016/j.jcp.2010.10.032, URL http://linkinghub.elsevier.com/retrieve/pii/S0021999110005887.

[7] Nguyen, N. C., and Peraire, J., "An Adaptive Shock-Capturing HDG Method for Compressible Flows," *20th AIAA Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2011, pp. AIAA 2011–3060. doi:10.2514/6.2011-3060, URL http://arc.aiaa.org/doi/abs/10.2514/6.2011-3060.

[8] Nguyen, N. C., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations," *J. Comput. Phys.*, Vol. 228, 2009, pp. 3232–3254.

[9] Nguyen, N. C., Peraire, J., and Cockburn, B., "An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection-diffusion equations," *J. Comput. Phys.*, Vol. 228, 2009, pp. 8841–8855.

[10] Nguyen, N. C., and Peraire, J., "Hybridizable discontinuous Galerkin methods for partial differential equations in continuum mechanics," *Journal of Computational Physics*, Vol. 231, No. 18, 2012, pp. 5955–5988. doi:10.1016/j.jcp.2012.02.033, URL http://linkinghub.elsevier.com/retrieve/pii/S0021999112001544.

[11] Nguyen, N. C., Peraire, J., and Cockburn, B., "A class of embedded discontinuous Galerkin methods for computational fluid dynamics," *Journal of Computational Physics*, Vol. 302, 2015, pp. 674–692. doi:10.1016/j.jcp.2015.09.024, URL http://www.sciencedirect.com/science/article/pii/S0021999115006178.

[12] Peraire, J., Nguyen, N. C., and Cockburn, B., "A hybridizable discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations," *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010, pp. AIAA 2010–363.

[13] Bassi, F., and Rebay, S., "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 131, 1997, pp. 267–279.

[14] Cockburn, B., and Shu, C.-W., "The local Discontinuous Galerkin Method for time-dependent convection-diffusion systems," *SIAM J. Numer. Anal.*, Vol. 35, 1998, pp. 2440–2463.

[15] Peraire, J., and Persson, P.-O., "The compact discontinuous Galerkin method for elliptic problems," *SIAM Journal on Scientific Computing*, Vol. 30, No. 4, 2008, pp. 1806–1824. doi:10.1137/070685518, URL http://link.aip.org/link/?SCE/30/1806/1.

[16] Persson, P.-O., and Peraire, J., "Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations," *SIAM Journal on Scientific Computing*, Vol. 30, No. 6, 2008, pp. 2709–2733. doi:10.1137/070692108, URL http://link.aip.org/link/?SCE/30/2709/1.

[17] Nguyen, N. C., and Peraire, J., "An efficient reduced-order modeling approach for non-linear parametrized partial differential equations," *International Journal for Numerical Methods in Engineering*, Vol. 76, No. 1, 2008, pp. 27–55. doi:10.1002/nme.2309, URL http://doi.wiley.com/10.1002/nme.2309.

[18] Fernandez, P., Nguyen, N. C., and Peraire, J., "A physics-based shock capturing method for unsteady laminar and turbulent flows," *56th AIAA Aerospace Sciences Meeting*, Orlando, Florida, 2018, pp. AIAA–2018–0062.

[19] Ciucă, C., Fernandez, P., Christophe, A., Nguyen, N., and Peraire, J., "Implicit hybridized discontinuous Galerkin methods for compressible magnetohydrodynamics," *Journal of Computational Physics: X*, 2019, p. 100042. doi:10.1016/J.JCPX.2019.100042, URL https://www.sciencedirect.com/science/article/pii/S2590055219300587.

[20] Moro, D., Nguyen, N. C., and Peraire, J., "Dilation-based shock capturing for high-order methods," *International Journal for Numerical Methods in Fluids*, Vol. 82, No. 7, 2016, pp. 398–416. doi:10.1002/fld.4223.

[21] Cockburn, B., and Shu, C.-W., "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463. doi:10.1137/S0036142997316712, URL http://epubs.siam.org/doi/abs/10.1137/S0036142997316712.

[22] Alexander, R., "Diagonally implicit Runge-Kutta methods for stiff ODEs," *SIAM J. Numer. Anal.*, Vol. 14, 1977, pp. 1006–1021.

[23] Marquardt, D. W., "An algorithm for least-squares estimation of nonlinear parameters," *SIAM*, Vol. 11, No. 2, 1963, pp. 431–441.

[24] Broyden, C. G., "The convergence of a class of double-rank minimization algorithms 1. General considerations," *IMA Journal of Applied Mathematics*, Vol. 6, No. 1, 1970, pp. 76–90. doi:10.1093/imamat/6.1.76, URL http://imamat.oxfordjournals.org/content/6/1/76.abstract.

[25] Fletcher, R., "A new approach to variable metric algorithms," *The Computer Journal*, Vol. 13, No. 3, 1970, pp. 317–322. doi:10.1093/comjnl/13.3.317, URL http://comjnl.oxfordjournals.org/content/13/3/317.short?rss=1{&}ssource=mfc.

[26] Goldfarb, D., "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, Vol. 24, No. 109, 1970, pp. 23–23. doi:10.1090/S0025-5718-1970-0258249-6, URL http://www.ams.org/mcom/1970-24-109/S0025-5718-1970-0258249-6/.

[27] Shanno, D. F., "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, Vol. 24, No. 111, 1970, pp. 647–647. doi:10.1090/S0025-5718-1970-0274029-X, URL http://www.ams.org/mcom/1970-24-111/S0025-5718-1970-0274029-X/.

[28] Gnoffo, P., and White, J., "Computational Aerothermodynamic Simulation Issues on Unstructured Grids," *37th AIAA Thermophysics Conference*, American Institute of Aeronautics and Astronautics, Portland, Oregon, 2004, pp. AIAA 2004–2371. doi:doi:10.2514/6.2004-2371, URL https://doi.org/10.2514/6.2004-2371.

[29] Wieting, A., and Holden, M., "Experimental shock-wave interference heating on a cylinder at Mach 6 and 8," *AIAA Journal*, Vol. 27, No. 11, 1989, pp. 1557–1565. doi:10.2514/3.10301, URL https://doi.org/10.2514/3.10301.

[30] Holden, M., "A study of flow separation in regions of shock wave-boundary layer interaction in hypersonic flow," *11th Fluid and PlasmaDynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 1978, pp. AIAA 1978–1169. doi:10.2514/6.1978-1169, URL http://arc.aiaa.org/doi/10.2514/6.1978-1169.

[31] Neuenhahn, T., and Olivier, H., "Influence of the wall temperature and the entropy layer effects on double wedge shock boundary layer interactions," *14th AIAA/AHI Space Planes and Hypersonic Systems and Technologies Conference*, Reston, Virigina, 2006, pp. AIAA 2006–8136. doi:10.2514/6.2006-8136, URL http://arc.aiaa.org/doi/10.2514/6.2006-8136.

[32] Schrijer, F., Scarano, F., van Oudheusden, B., and Bannink, W., "Application of PIV in a hypersonic double-ramp flow," *AIAA/CIRA 13th International Space Planes and Hypersonics Systems and Technologies Conference*, Reston, Virigina, 2005, pp. AIAA 2005–3331. doi:10.2514/6.2005-3331, URL http://arc.aiaa.org/doi/10.2514/6.2005-3331.

[33] Fedorov, A., "Transition and stability of high-speed boundary layers," *Annual Review of Fluid Mechanics*, Vol. 43, No. 1, 2011, pp. 79–95. doi:10.1146/annurev-fluid-122109-160750, URL http://www.annualreviews.org/doi/10.1146/annurev-fluid-122109-160750.

[34] Frauholz, S., Reinartz, B. U., Müller, S., and Behr, M., "Transition prediction for scramjets using γ-Reθt model coupled to two turbulence models," *Journal of Propulsion and Power*, Vol. 31, No. 5, 2015, pp. 1404–1422. doi:10.2514/1.B35630, URL http://arc.aiaa.org/doi/10.2514/1.B35630.